ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov395

Efficiency and Performance Trade-Offs: A Comparative Analysis of Statistical N-Gram Models and Resource-Optimized Small Language Models (SLMs) for Edge Computing Applications

Arnab Sen[1]

[1]Finance, Birla Institute of Technology and Science, Pilani, India

Publication Date: 2025/11/14

Abstract: Background: This research addresses the fundamental trade-off between model complexity and operational efficiency in Natural Language Processing (NLP), specifically for resource-constrained environments like edge computing.1 While Large Language Models (LLMs) offer unprecedented capabilities, their massive resource demands necessitate efficient alternatives.1 Materials and Methods: A critical comparative analysis was conducted on two dominant language model architectures: statistical N-gram models and modern Transformer-based Small Language Models (SLMs).1 The study evaluates their architectural mechanisms, efficiency metrics, tokenization strategies, and performance trade-offs, particularly focusing on metrics such as Perplexity (PPL) and qualitative semantic coherence.1 Results: SLMs, leveraging architectural optimizations like knowledge distillation and quantization, provide superior contextual understanding and deployment efficiency (days/weeks of training on small clusters) over N-gram models.1 N-gram models are severely limited by data sparsity, finite context windows, and storage bottlenecks, despite their fast lookup times.1 SLMs' use of subword tokenization (BPE) effectively eliminates the Out-of-Vocabulary (OOV) problem, preserving information lost by the N-gram's generic $\langle \text{unk} \rangle$ token.1 Conclusion: Resource-optimized SLMs are the most effective solution for high-performance, specialized NLP tasks in edge computing.1 While N-grams retain a niche as high-precision baselines for purely local statistical distributions, the efficiency and depth of comprehension favor the SLM for modern applications.[1]

Keyword: *Edge Computing; N-gram Models; Perplexity; Small Language Models; Transformer.*

How to Cite: Arnab Sen (2025). Efficiency and Performance Trade-Offs: A Comparative Analysis of Statistical N-Gram Models and Resource-Optimized Small Language Models (SLMs) for Edge Computing Applications.
International Journal of Innovative Science and Research Technology, 10(11), 363-369.
https://doi.org/10.38124/ijisrt/25nov395

I. INTRODUCTION

- *Background and Motivation for Language Modeling (LMs)*

Language Models (LMs) are foundational to Natural Language Processing (NLP), serving the core function of estimating the probability of word sequences.[1] Historically, LMs were indispensable components in classic NLP applications such as speech recognition and machine translation.[1] The field has since undergone a profound transformation, shifting from relying primarily on statistical methods to leveraging sophisticated deep learning architectures, most notably the Transformer.[1]

The subsequent emergence of Large Language Models (LLMs), characterized by their extensive parameter counts—often spanning hundreds of billions or even trillions—has unlocked unprecedented linguistic capabilities.[1] However, this scale introduces significant logistical and architectural challenges. The massive computational resources, protracted training times, and substantial energy consumption required to develop and operate LLMs necessitate infrastructure typically accessible only to the largest technology firms.[1] This resource exclusivity creates a significant economic and practical barrier to entry for many researchers and organizations. This reality necessitates the search for highly efficient, specialized, and practical alternatives that can operate effectively outside of vast cloud infrastructure.[1] The target operational parameters focus on resource-constrained environments, such as edge computing, mobile devices, and embedded systems, where speed, cost-effectiveness, and data privacy are paramount.[1] The analysis presented here is fundamentally filtered by the requirement of *deployability* in these constrained environments.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov395

➢ *Literature Context: The Shift from Statistical to Neural LMs*

For decades, statistical N-gram models were the dominant architecture for language processing.[1] These models relied on count-based probability estimation derived from empirical data, successfully providing a measurable metric for sequence likelihood.[1] Despite their historical importance, N-gram models suffer from a fundamental architectural limitation: a finite context window. They are unable to capture dependencies beyond the immediate $N-1$ preceding words.[1] This constraint, coupled with the exponential growth of computational demands and storage requirements as N increases, compelled the research community to seek more sophisticated, representation-based models.[1]

Initial attempts to capture longer-range dependencies included the introduction of Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs), which offered solutions to the vanishing gradient problem inherent in simple RNNs.[1] The definitive paradigm shift, however, was cemented by the introduction of the Transformer architecture in 2017.[1] Transformer-based architectures utilize self-attention mechanisms, processing the entire input sequence in parallel without recurrent units, which enables superior parallel processing and significantly better performance in capturing global context compared to their predecessors.[1] Small Language Models (SLMs) embody the practical realization of this architectural superiority, optimized for efficiency to bridge the capability gap between resource-intensive LLMs and demanding real-world, decentralized applications.[1] SLMs, therefore, represent not just a technical improvement but an *economic* solution to making advanced NLP accessible and deployable in latency-sensitive ecosystems like edge computing.[1]

➢ *Thesis Statement and Paper Contribution*

This research presents a critical comparative analysis of statistical N-gram models and modern Transformer-based Small Language Models (SLMs).[1] The objective is to rigorously evaluate their fundamental architectural mechanisms, resource efficiency metrics, and performance trade-offs under the severe constraints typical of edge computing deployment.[1] The core assertion of this paper is that resource-optimized Small Language Models (SLMs), utilizing modern architectural efficiencies, provide a superior solution for high-performance, specialized Natural Language Processing tasks when deployed under the resource constraints inherent to edge computing environments.[1]

This paper details how architectural differences—specifically in data handling, tokenization, and contextual awareness—dictate practical utility and define the ideal use case for each model type in the contemporary computational landscape.[1] The contribution lies in synthesizing the evidence to recommend the architecturally sound and deployable solution for decentralized AI applications.

II. MODEL ARCHITECTURES AND EFFICIENCY CONSIDERATIONS

This section provides a rigorous comparison of the model structures, focusing on how they manage memory, computation, and scaling, which are the primary determinants of feasibility in edge deployment.

➢ *N-gram Model Fundamentals and Sparsity Mitigation*

N-gram models operate by estimating the probability of the next word, w_i, given the preceding $N-1$ context words ($w_{i-N+1}...w_{i-1}$) using Maximum Likelihood Estimation (MLE).[1] This estimation is derived directly from empirical word count frequencies within a training corpus.[1] The selection of the context size, N, is a critical design choice, requiring a balance between the estimate's statistical stability and its appropriateness for the domain; for instance, trigrams ($N=3$) are often preferred for massive corpora, while bigrams ($N=2$) might suffice for smaller datasets.[1]

The primary architectural vulnerability of N-gram models is data sparsity. Since the number of possible N-grams grows exponentially with N, most potential sequences are never observed in the training data, leading to the "zero probability problem".[1] When this zero probability occurs, key evaluation metrics like perplexity cannot be calculated, demanding corrective action.[1] To address this, smoothing or discounting algorithms are mandatory; these techniques redistribute probability mass from frequent, observed events to unseen events, ensuring all plausible sequences have a non-zero probability.[1] While Laplace smoothing (add-one) is a simple baseline, it is insufficient for modern deployment due to its tendency to over-smooth the data.[1] Consequently, high-accuracy N-gram applications necessitate advanced techniques, such as Good–Turing discounting and Kneser-Ney smoothing, to manage this statistical necessity.[1]

The efficiency trade-off for N-grams is centered on storage versus lookup speed. While statistical lookups during inference are extremely fast, the models still represent a major resource bottleneck, particularly in large-scale applications.[1] The limitation is fixed memory size: the largest models can require storing hundreds of billions of N-grams and their associated count data, presenting a major infrastructure problem for decentralized deployment.[1] Research efforts to mitigate this focus on developing highly compact and fast-to-query implementations, often utilizing lossless compression techniques like tabular trie encodings, which can achieve representations as low as 23 bits per N-gram, significantly improving storage efficiency.[1]

➢ *Small Language Model (SLM) Architecture*

Small Language Models (SLMs) are compact versions of deep neural networks, defined typically as models ranging from 1 million to 10 billion parameters, fundamentally based on the Transformer architecture.[1] They are engineered specifically for efficiency and operational effectiveness in computationally limited settings.[1]

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov395

The architectural backbone of the SLM is the Transformer, which uses a multi-head self-attention mechanism to process the entire input sequence in parallel.[1] This parallel capacity allows the model to capture long-range dependencies efficiently, fundamentally overcoming the limited context window that constrains N-gram models.[1] Input text is converted into tokens, mapped to vectors via an Embedding layer, and processed through alternating attention and feedforward layers (Transformer blocks) to extract increasingly complex linguistic patterns.[1]

The designation "small" is achieved through a sophisticated optimization stack designed to reduce the model's size and computational footprint without incurring catastrophic performance degradation.[1] These architectural optimizations are crucial for meeting edge computing requirements. Key techniques include:

- *Knowledge Distillation:*
A smaller "student" model is trained to emulate the outputs and internal state representations of a larger, high-performing "teacher" model.[1] This process effectively compresses the complex knowledge base of the larger model into a lower parameter count.[1]

- *Pruning:*
Redundant or statistically insignificant parameters (weights and connections) within the neural network are systematically identified and removed, resulting in a thinned, faster network structure.[1]

- *Quantization:*
The precision of the numerical values used to store the model's weights is drastically reduced. For instance, converting weights from high-precision 32-bit floating-point numbers to lower-precision 8-bit integers substantially decreases the model's memory footprint and accelerates inference speed.[1] This process addresses the memory I/O bottleneck, as operations are performed on smaller, reduced-precision data types, decreasing memory bandwidth demands crucial for low-power edge deployment.

The critical distinction in resource constraint management is that N-grams face a challenge of **fixed memory size** that scales exponentially with vocabulary and context length, necessitating massive storage that fails the memory constraint for limited edge hardware.[1] SLMs, conversely, manage complexity during inference but solve the memory problem through density and quantization, allowing their dense vector set to be loaded and run efficiently.[1] By prioritizing speed and deployment efficiency, SLMs require significantly less infrastructure; their training process typically takes days to weeks on small GPU clusters, making them far more cost-effective and accessible than LLMs.[1]

III. PREPROCESSING AND TOKENIZATION IMPACT ON MODEL PERFORMANCE

The method by which raw text is converted into processing units (tokenization) dictates a model's ability to handle the linguistic nuances of complexity, novelty, and compositionality.[1]

➢ *Tokenization for N-gram Models*

N-gram models traditionally rely on simple word-level tokenization, often splitting text based on white spaces, a method that immediately introduces struggles with boundary issues such as attached punctuation.[1] The more profound challenge is the management of Out-of-Vocabulary (OOV) words—those present in the test corpus but absent from the training lexicon.[1] OOV words instantly result in a zero-probability scenario, halting the generative process.[1]

The standard protocol for N-gram models addresses OOV by introducing a special token, typically denoted as $\langle \text{unk} \rangle$ (unknown).[1] All rare or OOV words encountered during training are effectively replaced with this single token before the N-gram counts are accumulated.[1] While this necessity strongly influences performance metrics like perplexity, the methodology imposes a significant structural limitation: it inherently destroys local contextual information.[1] When complex or rare words are collapsed into a generic $\langle \text{unk} \rangle$ representation, all specific linguistic information about that term, including its morphology or domain relevance, is lost.[1] This lack of informational integrity compromises the statistical model's ability to predict stably and richly when encountering linguistic inputs outside of the previously observed training corpus.[1]

➢ *Subword Tokenization Strategies for SLMs*

Transformer-based SLMs utilize sophisticated subword tokenization methods to mitigate the OOV problem and optimize vocabulary efficiency, achieving a balance between expressiveness and memory limits.[1] This capability is a critical architectural advantage, allowing SLMs to manage diverse inputs without the severe contextual compromises faced by count-based N-gram models.[1]

The primary methods employed include:

- *Byte Pair Encoding (BPE):*
BPE is a frequency-based, deterministic approach.[1] It initiates with a base vocabulary of individual characters and iteratively merges the most frequent adjacent symbol pairs into longer, meaningful tokens until a predefined vocabulary size is reached.[1] This mechanism ensures that any rare or unseen word can always be reliably decomposed into known subword components, thereby eliminating the hard OOV problem.[1]

- *Unigram and WordPiece:*
WordPiece, often used in large models, evaluates the statistical gain of merging two symbols to ensure efficiency.[1] The Unigram approach begins with a comprehensive vocabulary and statistically trims it down based on

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov395

probability.[1]

The shift to subword tokenization provides SLMs with an information-preserving foundation for context processing.[1] For example, if an N-gram encounters a novel, complex term like "re-quantization," it is forced to map it to $\langle \text{unk} \rangle$, resulting in a loss of all semantic content. Conversely, a BPE-based SLM will tokenize the same word into its constituent semantic components (e.g., "re-", "quant", "ization"), allowing the model to leverage the trained embeddings of those components.[1] This capacity for generalization, inferring meaning from morphological components, ensures that the SLM maintains contextual information and consistency, a vital benefit when processing domain-specific or novel text in edge applications.[1] This allows SLMs to achieve linguistic generalization, whereas the $\langle \text{unk} \rangle$ token forces N-grams into a mode of **memorization**, fundamentally restricting their utility in processing dynamic, modern language.[1]

IV. EVALUATION AND EXPERIMENTAL SETUP

A comprehensive comparison of LMs requires a dual evaluation approach, moving beyond simple intrinsic measurements to incorporate extrinsic, task-specific metrics that accurately assess quality and utility.[1]

➢ *Intrinsic Metric: Perplexity (PPL) Analysis*

Perplexity (PPL) is the standard intrinsic metric for evaluating language models.[1] Mathematically, it is defined as the inverse probability of the test set, normalized by the number of words or tokens, which allows for comparison across texts of varying lengths.[1] A superior model assigns a higher probability to the test corpus, indicating it is less "surprised" by the sequence of words, thereby yielding a lower perplexity score.[1] Numerically, PPL can be interpreted as the average number of equally likely choices the model considers for each word.[1]

However, PPL faces significant methodological constraints, particularly when comparing different architectures.[1] Critically, PPL computation is impossible if the model assigns a zero probability to any sequence in the test set, which underscores why smoothing is not merely optional but mandatory for N-gram models.[1] Furthermore, PPL scores are highly sensitive to methodological choices concerning OOV word handling and the total vocabulary size employed.[1] Interpreting a PPL score between a context-limited, count-based N-gram and a deep, representation-based SLM can be misleading.[1] While N-grams, optimized with advanced smoothing, may achieve a decent PPL score, this result primarily reflects localized statistical fluency, not the quality of long-term generative coherence.[1]

➢ *Extrinsic and Qualitative Performance Metrics*

Since the utility of LMs, particularly SLMs, often resides in their ability to perform high-level, specialized tasks, relying solely on intrinsic measures such as PPL is insufficient.[1] Extrinsic and qualitative metrics are essential for assessing the generated output quality, particularly concerning coherence and relevance.[1]

For structured tasks where a clear human reference text is available (e.g., machine translation), Lexical Overlap Metrics are commonly used.[1] Metrics such as BLEU (Bilingual Evaluation Understudy) and ROUGE quantify the similarity between the generated text and the reference by calculating the overlap of N-grams.[1] These metrics, however, can be restrictive because they penalize outputs that use semantically equivalent synonyms or alternative sentence structures, indicating a failure to capture deeper linguistic variation.[1]

To capture the profound linguistic qualities of generative models, Semantic and Coherence Metrics are required, reflecting the architectural strengths of SLMs.[1] These include:

- *Fluency:*
  Evaluation of how natural, grammatical, and easy to read the generated text is, adhering to the conventions of the target language.[1]

- *Coherence:*
  Assessment of the logical and thematic consistency of the generated text, ensuring a smooth and unified flow of ideas.[1]

- *Relevance:*
  Measurement of the degree to which the generated output accurately addresses or aligns with the initial input prompt or specified task.[1]

A superior evaluation tool must align structurally with the modern model architecture. For SLMs, which are semantic and vector-driven, metrics that transcend simple word overlap are necessary. Embedding-Based Metrics, such as BERTScore, are vital because they leverage dense vector representations (embeddings) from Transformer models to measure semantic similarity between the generated and reference texts.[1] This approach provides a more accurate assessment of conceptual similarity, which is crucial for evaluating the nuanced, context-aware output of SLMs.[1]

The following table summarizes the comparative landscape of evaluation metrics necessary for assessing language models in resource-constrained environments:

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov395

Table 1 Comparative Analysis of Language Model Evaluation Metrics for Resource-Constrained Environments

Metric Category	Example Metrics	Targeted Assessment	Applicability
Language Modeling (Intrinsic)	Perplexity (PPL) \cite{26}	Statistical fluency, predictive power of next token	Foundational for both; highly sensitive to smoothing/OOV handling in N-grams [1]
Lexical Overlap (Extrinsic)	BLEU, ROUGE \cite{32}	N-gram match to reference text	Useful for structured tasks; limited in capturing semantic diversity [1]
Qualitative/Semantic	Coherence, Fluency, BERTScore \cite{32}	Logical flow, semantic similarity, grammatical correctness	Essential for evaluating the generative quality of SLMs, which excel at long-range context [1]

V. DISCUSSION AND PRACTICAL IMPLICATIONS

➢ *Trade-off Analysis: Resource Consumption vs. Accuracy*

The core distinction between N-gram models and SLMs is the balance they strike between computational cost, resource efficiency, and linguistic fidelity.[1] SLMs offer a decisive advantage in deployment efficiency: they are significantly faster to deploy and fine-tune, typically requiring only days or weeks on smaller GPU clusters, thereby making them financially and logistically accessible to a wider range of institutions.[1] By prioritizing efficiency through architectural optimizations—specifically pruning, knowledge distillation, and aggressive quantization—SLMs deliver high performance suitable for resource-constrained environments like embedded systems, mobile devices, and IoT applications.[1] They enable the real-time interaction that massive, high-latency LLMs cannot sustain in these settings.[1]

Conversely, while N-gram models are inherently fast during the immediate query execution phase, they are structurally hampered by storage constraints. Their reliance on massive count data makes them prone to storage bottlenecks, often requiring complex, time-consuming implementation research focused on advanced compression techniques and trie-based encoding to achieve practical deployment sizes.[1] Furthermore, this reliance on large data structures introduces an additional constraint in low-power environments: the massive I/O required to manage and query these count tables, even when compressed, translates directly into increased power draw and startup latency. The SLM's small, dense, quantized memory model is therefore superior for energy-efficient, long-term edge deployment. Most significantly, the N-gram architecture's inherent inability to capture dependencies beyond the limited N-gram window fundamentally limits its potential for complex, coherent text generation, rendering it obsolete for modern generative NLP tasks.[1]

➢ *Niche Advantages of N-gram Models*

Although SLMs represent a profound architectural advancement, statistical N-gram models have not become entirely obsolete.[1] The research suggests that the powerful inductive bias of Transformer-based models, which is inherently optimized for learning complex, dense semantic representations, can paradoxically hinder their performance when required to approximate simple, non-representational probability distributions.[1] For instance, when tested on constrained language models with arbitrary next-symbol probabilities—distributions that N-gram models inherently capture perfectly through raw counting—transformers have been observed to perform comparatively worse than the count-based techniques.[1]

This suggests that the N-gram's retention in the field is not a measure of architectural strength for general NLP, but rather a reflection of their utility in covering statistical blind spots of the Transformer architecture. N-gram models may still serve as highly effective, high-precision baselines for extremely specialized, vocabulary-limited sequence prediction tasks where computational simplicity and a strict reliance on purely local statistics are the design requirement.[1] The N-gram architecture's structural simplicity, when combined with advanced statistical correction like Kneser-Ney smoothing, grants it a specific, tactical operational utility for niche baseline comparisons.[1]

➢ *Real-World SLM Applications and Specialization*

SLMs excel precisely because they are focused on specialization.[1] Their success comes from being optimized for targeted tasks rather than general conversational breadth. They are finding application across various latency-critical and regulated domains, including specialized legal document assistance, customized workflows in finance, and on-device symptom checking in healthcare.[1]

This necessary focus on efficiency, however, imposes a constraint: SLMs are typically characterized by a narrow focus, optimized explicitly for the task for which they were fine-tuned.[1] A model trained as a specialized legal document assistant, for example, is not expected to perform well at complex code generation or casual conversation unless further, task-specific fine-tuning is performed.[1] This narrow specialization means SLMs do not possess the broad, zero-shot generalization capabilities of massive LLMs.[1] Nevertheless, for the targeted enterprise use case, this specialization delivers superior accuracy, lower latency, and faster development cycles, making them the pragmatic and responsible choice for edge computing.[1] Crucially, the relative cost-effectiveness of SLM development democratizes advanced NLP, allowing smaller organizations to deploy competitive, task-specific AI solutions without requiring access to hyperscale infrastructure.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov395

➢ *Future Directions: SLM/LLM Collaboration*

The future trajectory of language modeling is increasingly trending toward collaboration and optimization across architectural scales, rather than strict competition.[1] SLMs are emerging as essential tools for optimizing the LLM ecosystem itself.[1] They can be strategically leveraged to generate sizable, high-quality training datasets, significantly reducing the reliance on laborious and expensive manual annotation during the training and fine-tuning phases of massive LLMs.[1]

Furthermore, utilizing SLMs as highly specialized agents, data routers, or task orchestrators can mitigate the necessity for constant, full-scale fine-tuning or querying of massive LLMs.[1] This cooperative approach leads to substantial reductions in required computational resources and helps address critical concerns related to scalability, data privacy, and ethical safety often associated with deploying the largest foundation models.[1]

VI. CONCLUSION AND FUTURE WORK

➢ *Conclusion*

This comparative analysis confirms that resource-optimized Small Language Models (SLMs) represent the most effective and architecturally feasible solution for high-performance natural language tasks deployed in edge computing and resource-constrained environments.[1] The Transformer architecture, optimized through industrial techniques such as knowledge distillation, pruning, and quantization, grants SLMs superior contextual understanding, effectively addressing the fundamental limitations of long-range dependency failure and the limited context window inherent in statistical N-gram models.[1]

Moreover, the use of Byte Pair Encoding (BPE) subword tokenization by SLMs provides a robust, information-preserving solution for Out-of-Vocabulary words, crucially avoiding the substantial information loss associated with the N-gram's reliance on the generic $\langle \text{unk} \rangle$ token approach.[1] While highly optimized N-gram models can still offer fast, compact, and high-precision baselines for specific, localized probability distributions, their severe storage constraints and inherent data sparsity issues relegate them to niche or legacy applications.[1] Ultimately, the selection between these two architectures is dictated by the available resource budget and the required scope of linguistic generalization; the confluence of efficiency, depth of comprehension, and operational feasibility decisively favors the SLM architecture for modern decentralized applications.[1]

➢ *Future Work*

Future research must transition from theoretical architectural comparisons to rigorous empirical implementation and verification within simulated edge computing parameters.[1] This necessitates conducting quantitative side-by-side performance tests of a state-of-the-art compressed N-gram model (employing advanced smoothing such as Kneser-Ney) against a highly pruned and aggressively quantized SLM (e.g., a small-scale decoder-only Transformer).[1]

These experiments must integrate a comprehensive suite of metrics, balancing the traditional intrinsic measure of Perplexity with crucial semantic measures, including Coherence and BERTScore, to accurately quantify the genuine performance trade-off relative to the architectural resource footprint.[1] Additionally, focused efforts should be directed toward refining distillation and pruning methodologies to maximize the knowledge transfer effectiveness from LLMs to SLMs, specifically optimizing the resultant models for ultra-low-power, highly specialized domain deployment efficiency.[1]

REFRENCES

[1]. Plagiarism Free Writing Techniques: Avoiding Common Pitfalls in Research Writing - San Francisco Edit, accessed on November 7, 2025, https://www.sfedit.net/plagiarism-free-writing-techniques-avoiding-common-pitfalls-in-research-writing/

[2]. How to Write a Plagiarism-Free Research Paper or Thesis - Papergen AI, accessed on November 7, 2025, https://www.papergen.ai/blog/how-to-write-a-plagiarism-free-research-paper-or-thesis

[3]. How to Avoid Plagiarism | Harvard Guide to Using Sources, accessed on November 7, 2025, https://usingsources.fas.harvard.edu/how-avoid-plagiarism-0

[4]. Best Practices to Avoid Plagiarism - Purdue OWL, accessed on November 7, 2025, https://owl.purdue.edu/owl/avoiding_plagiarism/best_practices.html

[5]. IOSR Manuscript Preparation Guidelines | PDF - Scribd, accessed on November 7, 2025, https://www.scribd.com/document/768600584/IOSR-Manuscript-Preparation-Guidelines

[6]. Paper preparation guidelines for IOSR Journal of Engineering, accessed on November 7, 2025, https://ternaengg.ac.in/equinox2018/PaperFormat.pdf

[7]. Manuscript Preparation Guidelines (2 Page) | PDF | Abstract (Summary) | Paragraph - Scribd, accessed on November 7, 2025, https://www.scribd.com/document/98842041/Manuscript-Preparation-Guidelines-2-Page

[8]. IOSR Journal of Computer Engineering (IOSR-JCE) Template - International Organization of Scientific Research - SciSpace, accessed on November 7, 2025, https://scispace.com/formats/international-organization-of-scientific-research/iosr-journal-of-computer-engineering-iosr-jce/489e0da8074e4cfc8b861a6709e6969f

[9]. Paper Template - IOSR Journal, accessed on November 7, 2025, https://www.iosrjournals.org/doc/Paper%20Template.doc

[10]. N-gram Language Models - Stanford University, accessed on November 7, 2025, https://web.stanford.edu/~jurafsky/slp3/3.pdf

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov395

[11]. Word n-gram language model - Wikipedia, accessed on November 7, 2025, https://en.wikipedia.org/wiki/Word_n-gram_language_model
[12]. Transformer (deep learning architecture) - Wikipedia, accessed on November 7, 2025, https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture)
[13]. Small Language Models (SLM): A Comprehensive Overview - Hugging Face, accessed on November 7, 2025, https://huggingface.co/blog/jjokah/small-language-model
[14]. SLM vs LLM: The Key Differences - WEKA, accessed on November 7, 2025, https://www.weka.io/learn/ai-ml/slm-vs-llm/
[15]. What Are Small Language Models (SLMs)? A Practical Guide - Aisera, accessed on November 7, 2025, https://aisera.com/blog/small-language-models/
[16]. Large and small language models: A side-by-side comparison - Rabiloo, accessed on November 7, 2025, https://rabiloo.com/blog/large-and-small-language-models-a-side-by-side-comparison
[17]. Understanding Language Modeling: From N-grams to Transformer-based Neural Models | by Roshmita Dey | Medium, accessed on November 7, 2025, https://medium.com/@roshmitadey/understanding-language-modeling-from-n-grams-to-transformer-based-neural-models-d2bdf1532c6d
[18]. LLM Transformer Model Visually Explained - Polo Club of Data Science, accessed on November 7, 2025, https://poloclub.github.io/transformer-explainer/
[19]. Comparing the Effect of Smoothing and N-gram Order - Scholarship Repository @ Florida Tech, accessed on November 7, 2025, https://repository.fit.edu/cgi/viewcontent.cgi?article=1712&context=etd
[20]. Faster and Smaller N-Gram Language Models - ACL Anthology, accessed on November 7, 2025, https://aclanthology.org/P11-1027.pdf
[21]. Faster and Smaller N-Gram Language Models - The Berkeley NLP Group, accessed on November 7, 2025, http://nlp.cs.berkeley.edu/pubs/Pauls-Klein_2011_LM_paper.pdf
[22]. Summary of the tokenizers - Hugging Face, accessed on November 7, 2025, https://huggingface.co/docs/transformers/en/tokenizer_summary
[23]. Predictive Incremental Parsing Helps Language Modeling - ACL Anthology, accessed on November 7, 2025, https://aclanthology.org/C16-1026.pdf
[24]. Byte Pair Encoding vs. Unigram Tokenization: A Deep Dive into Subword Models - Medium, accessed on November 7, 2025, https://medium.com/@hexiangnan/byte-pair-encoding-vs-unigram-tokenization-a-deep-dive-into-subword-models-4963246e9a34
[25]. Two minutes NLP — A Taxonomy of Tokenization Methods | by Fabio Chiusano - Medium, accessed on November 7, 2025, https://medium.com/nlplanet/two-minutes-nlp-a-taxonomy-of-tokenization-methods-60e330aacad3
[26]. Arnab Sen Paper.docx
[27]. Can Transformers Learn n-gram Language Models? - ACL Anthology, accessed on November 7, 2025, https://aclanthology.org/2024.emnlp-main.550.pdf
[28]. A Comparison of Tokenization Impact in Attention Based and State Space Genomic Language Models | bioRxiv, accessed on November 7, 2025, https://www.biorxiv.org/content/10.1101/2024.09.09.612081v2.full-text
[29]. A Comparative analysis of different LLM Evaluation Metrics | by Satyadeep Behera - Medium, accessed on November 7, 2025, https://medium.com/@satyadeepbehera/a-comparative-analysis-of-different-llm-evaluation-metrics-98395c3d8e79
[30]. Perplexity Metric for LLM Evaluation - Analytics Vidhya, accessed on November 7, 2025, https://www.analyticsvidhya.com/blog/2025/04/perplexity-metric-for-llm-evaluation/
[31]. How to evaluate a text generation model: strengths and limitations of popular evaluation metrics - The Analytics Lab, accessed on November 7, 2025, https://theanalyticslab.nl/how-to-evaluate-a-text-generation-model-strengths-and-limitations-of-popular-evaluation-metrics/
[32]. LLM Evaluation: 15 Metrics You Need to Know, accessed on November 7, 2025, https://arya.ai/blog/llm-evaluation-metrics
[33]. Testing & Evaluating Large Language Models(LLMs): Key Metrics and Best Practices Part-2, accessed on November 7, 2025, https://medium.com/@sumit.somanchd/testing-evaluating-large-language-models-llms-key-metrics-and-best-practices-part-2-0ac7092c9776
[34]. Small Language Models: A Business Leader's Guide to Affordable, Task-Tuned AI, accessed on November 7, 2025, https://deliveringdataanalytics.com/small-language-models-business-guide/
[35]. The Rise of Small Language Models - IEEE Computer Society, accessed on November 7, 2025, https://www.computer.org/csdl/magazine/ex/2025/01/10897262/24uGPS4TUQ0